UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/526,700	03/04/2005	Shridhar Mubaraq Mishra	1890-0207	7455

50255 7590 04/15/2009

MAGINOT, MOOR & BECK
111 MONUMENT CIRCLE, SUITE 3000
BANK ONE CENTER/TOWER
INDIANAPOLIS, IN 46204

EXAMINER	
HICKS, MICHAEL J	
ART UNIT	PAPER NUMBER
2165	

MAIL DATE	DELIVERY MODE
04/15/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/526,700	MISHRA ET AL.	
	Examiner	Art Unit	
	Michael J. Hicks	2165	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 August 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *12-31* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *12-31* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *04 March 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 12-31 Pending.

Claims 1-11 Canceled.

Response to Arguments

2. Examiner apologizes for the oversight in the last office action noted by Applicants in the Communication filed 8/19/2008. A corrected office action considering Claims 12-31 as filed in the amended claim listing on 3/4/2005 is included herein.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

4. Claims 12-31 rejected under 35 U.S.C. 102(e) as being anticipated by Lu et al.

(U.S. Patent Number 677984, filed 10/23/2000, Patented 8/17/2004 and referred to hereinafter as Lu).

As per Claim 12, Lu discloses a method of comparing a data key to a rule, the method including: dividing the data key into a plurality of chunks (i.e. *"Accordingly, one object of the present invention is to provide a flexible and high-performance packet classification algorithm that involves the conversion of original rule database into a rule mapping table format for*

storage. The rule mapping table is formed by dividing an input key into a plurality of sub-keys, and then sequentially comparing the ordering of each sub-key with the same sub-key field of each rule. Finally, the results of the comparison (`1` indicates a match while a `0` indicates a mismatch) are stored in the rule mapping table through bit mapping." The preceding text excerpt clearly indicates that a data key is divided into a plurality of chunks (e.g. subkeys).) (Column 2, Lines 20-31), extracting data from a memory at an address corresponding to the value of each of at least some of the chunks, the data being previously stored at the address of the memory such that the data has a value corresponding to whether a bitwise comparison of the chunk of the corresponding data key with a portion of the mask is equal to a bitwise comparison of the portion of the mask and a corresponding portion of the rule (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120, searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1 }. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150."* The preceding text excerpt clearly indicates that a rule table, which indicates correspondence between the rules and the keys, is built by comparing the subkeys with masked portions of the rule. Examiner notes that the comparison is a

bitwise AND operation.) (Column 4, Lines 33-57); and examining the data extracted from each memory address corresponding to the at least some of the chunks to determine if the rule is obeyed for the entire data key (i.e. *"The goal of this invention is to provide a high-speed and economical search method particularly for searching data that includes don't care bit fields. To simplify explanation, a rule table having just five rules is selected in the following illustration. FIG. 1 is a diagram showing a packet classification database having five 8-bit rules. As shown in FIG. 1, the rule table 100 has five 8-bit rules with each rule bit having `1`, `0` or `x` (don't care). The search method of this invention is capable of finding a rule that conforms to a particular input key in rule table 100. When more than one rule conforms to the input key, the search method takes up the earlier one in the rule table list as the lookup result of the search. Using an input key #0(`111110011`) as an example, conformed rules in table 100 includes rule #1(`11110×11`), rule #2(`1×××0011` and rule #4(`××××××××`). Under such circumstances, the search method chooses rule #1 as a search result 100. The following is a description of the packet classification algorithm provided by this invention. First, the original rule table must be converted into a rule mapping table and stored in a memory unit. FIG. 2 is a diagram showing the packet classification algorithm according to this invention. The mapping table 120 shown in FIG. 2 is generated by dividing the input key into a plurality of sub-keys. Thereafter, various combinations of the bits inside each sub-key value and same sub-key value field to of each rule are compared. The result of the comparison is stored in mapping table 120 according to a bit map method (`1` represents a match while `0` represent a mismatch). Later, in the following description, rule vector (I, J) is used to refer to the converted bit mapping of mapping table 120 when the value of the Ith sub-key with sub-key value equal to J."* The preceding text excerpt clearly indicates that that data key is determined to obey the rule if all of the subkeys match each of the corresponding rule vectors. Examiner notes that this correspondence data is stored in the rule mapping table and examined to determine if keys conform to rules.) (Column 4, Lines 1-32).

As per Claim 13, Lu discloses the memory is structured into chunks corresponding to the chunks of the data key, the chunks of the memory being grouped into sections, and wherein the method further comprises simultaneously for different sections of the memory successively extracting the data for the corresponding chunks of memory within each section (i.e. *"Through the said technique, rule mapping table lookup can be achieved with each search engine processing a portion of the sub-key fields in parallel. By using interleave matrix, the search algorithm is even capable of processing rules that have a varying width. Furthermore, this invention can support a plurality of rule databases or sub-tables. The only criteria are to set up the initial scan value, the terminal scan value and rule width of the desired search sub-table. After the setting of the initial scan value, the search engines can search for the sub-table automatically. Consequently, a plurality of rule databases each having a different length and width can coexist inside the same search engine so that operating characteristics (such as speed and volume occupation) and flexibility (the coexistence of different rule databases) are improved."* The preceding text excerpt clearly indicates that the subkeys are processed in parallel.) (Column 6, Lines 21-35).

As per Claim 14, Lu discloses address from where the data is extracted data corresponds to chunk of the data key (i.e. Figure 2 clearly indicates that the address in the mapping table from which the mapping data is extracted corresponds to the chunk of the data key.) (Figure 2).

As per Claim 15, Lu discloses a method of comparing a data key to portions of a plurality of rules, the portions being defined by corresponding masks, the method including: dividing the data key into chunks (i.e. *"Accordingly, one object of the present invention is to provide a flexible and high-performance packet classification algorithm that involves the conversion*

of original rule database into a rule mapping table format for storage. The rule mapping table is formed by dividing an input key into a plurality of sub-keys, and then sequentially comparing the ordering of each sub-key with the same sub-key field of each rule. Finally, the results of the comparison (`1` indicates a match while a `0` indicates a mismatch) are stored in the rule mapping table through bit mapping." The preceding text excerpt clearly indicates that a data key is divided into a plurality of chunks (e.g. subkeys).) (Column 2, Lines 20-31), successively for each of said rules: (i) extracting data from an address obtained from each of the chunks of the data key, the data being stored at the address of the memory such that the data has a value corresponding to whether a bitwise comparison of the chunk of the corresponding data key with a portion of the mask is equal to a bitwise comparison of the portion of the mask and a corresponding portion of the rule (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120, searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1 }. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150."* The preceding text excerpt clearly indicates that a rule table, which indicates correspondence between the rules and the keys, is built by comparing the subkeys with masked portions

of the rule. Examiner notes that the comparison is a bitwise AND operation.) (Column 4, Lines 33-57); and (ii) examining the data extracted from each of the chunks of the data key to determine if the rule is obeyed for the entire data key (i.e. *"The goal of this invention is to provide a high-speed and economical search method particularly for searching data that includes don't care bit fields. To simplify explanation, a rule table having just five rules is selected in the following illustration. FIG. 1 is a diagram showing a packet classification database having five 8-bit rules. As shown in FIG. 1, the rule table 100 has five 8-bit rules with each rule bit having `1`, `0` or `x` (don't care). The search method of this invention is capable of finding a rule that conforms to a particular input key in rule table 100. When more than one rule conforms to the input key, the search method takes up the earlier one in the rule table list as the lookup result of the search. Using an input key #0(`111110011`) as an example, conformed rules in table 100 includes rule #1(`11110×11`), rule #2(`1×××0011` and rule #4(`××××××××`). Under such circumstances, the search method chooses rule #1 as a search result 100. The following is a description of the packet classification algorithm provided by this invention. First, the original rule table must be converted into a rule mapping table and stored in a memory unit. FIG. 2 is a diagram showing the packet classification algorithm according to this invention. The mapping table 120 shown in FIG. 2 is generated by dividing the input key into a plurality of sub-keys. Thereafter, various combinations of the bits inside each sub-key value and same sub-key value field to of each rule are compared. The result of the comparison is stored in mapping table 120 according to a bit map method (`1` represents a match while `0` represent a mismatch). Later, in the following description, rule vector (I, J) is used to refer to the converted bit mapping of mapping table 120 when the value of the Ith sub-key with sub-key value equal to J."* The preceding text excerpt clearly indicates that that data key is determined to obey the rule if all of the subkeys match each of the corresponding rule vectors. Examiner notes that this correspondence data is stored in the rule mapping table and examined to determine if keys conform to rules.) (Column 4, Lines 1-32).

As per Claim 16, Lu discloses the memory is structured two dimensionally, a first dimension corresponding to the chunks of the data key and a second dimension corresponding to the different rules (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120, searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1 }. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150."* The preceding text excerpt and Figure 2 clearly indicate that the mapping table and the rule vectors are stored in two separate data dimensions.) (Figure 2; Column 4, Lines 33-57).

As per Claim 17, Lu discloses examining the data further comprises performing at least one AND operation on the data extracted from each of the chunks to determine if the rule is obeyed for the entire data key (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key*

is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120, searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1 }. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150." The preceding text excerpt clearly indicates that a rule table, which indicates correspondence between the rules and the keys, is built by comparing the subkeys with masked portions of the rule. Examiner notes that the comparison is a bitwise AND operation.) (Column 4, Lines 33-57).

As per Claim 18, Lu discloses a system for comparing a data key to portions of a plurality of rules defined by corresponding masks, the system including: an interface configured to receive the data key, and to divide the data key into chunks (i.e. *"Accordingly, one object of the present invention is to provide a flexible and high-performance packet classification algorithm that involves the conversion of original rule database into a rule mapping table format for storage. The rule mapping table is formed by dividing an input key into a plurality of sub-keys, and then sequentially comparing the ordering of each sub-key with the same sub-key field of each rule. Finally, the results of the comparison (`1` indicates a match while a `0` indicates a mismatch) are stored in the rule mapping table through bit mapping."* The preceding text excerpt clearly indicates that a data key is divided into a plurality of chunks (e.g. subkeys).) (Column 2, Lines 20-31), a memory configured to receive the chunks of the data key from the interface, and for successive

rules, to use the chunks of the data key and the rule as address data to extract data indicative of whether a bitwise comparison of each chunk of the data key with a corresponding portion of the mask is equal to a bitwise comparison of the corresponding portion of the mask and the corresponding portion of the rule (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120, searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1 }. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150."* The preceding text excerpt clearly indicates that a rule table, which indicates correspondence between the rules and the keys, is built by comparing the subkeys with masked portions of the rule. Examiner notes that the comparison is a bitwise AND operation.) (Column 4, Lines 33-57); and a comparator for examining the data extracted from the memory to determine if the rules are obeyed for the entire data key (i.e. *"The goal of this invention is to provide a high-speed and economical search method particularly for searching data that includes don't care bit fields. To simplify explanation, a rule table having just five rules is selected in the following illustration. FIG. 1 is a diagram showing a packet classification database having five 8-bit rules. As shown in FIG. 1, the rule table 100 has five 8-bit rules with each rule bit having `1`, `0` or `x` (don't care). The search method of this invention is capable of finding a rule that conforms to*

a particular input key in rule table 100. When more than one rule conforms to the input key, the search method takes up the earlier one in the rule table list as the lookup result of the search. Using an input key #0(`111110011`) as an example, conformed rules in table 100 includes rule #1(`11110×11`), rule #2(`1×××0011` and rule #4(`××××××××`). Under such circumstances, the search method chooses rule #1 as a search result 100. The following is a description of the packet classification algorithm provided by this invention. First, the original rule table must be converted into a rule mapping table and stored in a memory unit. FIG. 2 is a diagram showing the packet classification algorithm according to this invention. The mapping table 120 shown in FIG. 2 is generated by dividing the input key into a plurality of sub-keys. Thereafter, various combinations of the bits inside each sub-key value and same sub-key value field to of each rule are compared. The result of the comparison is stored in mapping table 120 according to a bit map method (`1` represents a match while `0` represent a mismatch). Later, in the following description, rule vector (I, J) is used to refer to the converted bit mapping of mapping table 120 when the value of the Ith sub-key with sub-key value equal to J." The preceding text excerpt clearly indicates that that data key is determined to obey the rule if all of the subkeys match each of the corresponding rule vectors. Examiner notes that this correspondence data is stored in the rule mapping table and examined to determine if keys conform to rules.) (Column 4, Lines 1-32).

As per Claim 19, Lu discloses the memory is structured two dimensionally, with a first dimension corresponding to the different chunks of the data key and a second dimension corresponding to the different rules (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120,*

searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1}. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150." The preceding text excerpt and Figure 2 clearly indicate that the mapping table and the rule vectors are stored in two separate data dimensions.) (Figure 2; Column 4, Lines 33-57).

As per Claim 20, Lu discloses the memory comprises a plurality of memory devices, each memory device corresponding to multiple chunks of the data key, such that all of the memory devices together correspond to all of the chunks of the data key (i.e. *"Furthermore, the rule mapping table can be dissected into a plurality of sub-tables such that the number of rules and rule width in each sub-table can be set. Each sub-table has an initial scan value register, a terminal scan value register and a register for recording the width of the sub-table. Each sub-table can even have a register for registering the initial address of memory for holding associated data and a register for registering size of storage location for the associated data."* The preceding text excerpt along with Figure 4 clearly indicates that the mapping tables, which hold the mappings corresponding to multiple chunks of the data key, may be divided into subtables which contain multiple registers. Examiner notes that both the subtables and the registers may be considered to be memory devise and that all the subtables together will correspond to all of the chunks of the data key.) (Figure 4, Column3, Lines 9-17).

As per Claim 21, Lu discloses the rules are divided into groups of rules, and wherein each memory device stores the data corresponding to one of the groups of the

rules (i.e. *"Through the said technique, rule mapping table lookup can be achieved with each search engine processing a portion of the sub-key fields in parallel. By using interleave matrix, the search algorithm is even capable of processing rules that have a varying width. Furthermore, this invention can support a plurality of rule databases or sub-tables. The only criteria are to set up the initial scan value, the terminal scan value and rule width of the desired search sub-table. After the setting of the initial scan value, the search engines can search for the sub-table automatically. Consequently, a plurality of rule databases each having a different length and width can coexist inside the same search engine so that operating characteristics (such as speed and volume occupation) and flexibility (the coexistence of different rule databases) are improved."* The preceding text excerpt clearly indicates that groupings of differently sized rules are grouped together and stored in separate databases (e.g. memory devices).) (Column 6, Lines 21-35).

As per Claim 22, Lu discloses the memory further comprises a plurality of memory devices, the rules are divided into groups of rules, and each memory device stores the data corresponding to one of the groups of the rules (i.e. *"Through the said technique, rule mapping table lookup can be achieved with each search engine processing a portion of the sub-key fields in parallel. By using interleave matrix, the search algorithm is even capable of processing rules that have a varying width. Furthermore, this invention can support a plurality of rule databases or sub-tables. The only criteria are to set up the initial scan value, the terminal scan value and rule width of the desired search sub-table. After the setting of the initial scan value, the search engines can search for the sub-table automatically. Consequently, a plurality of rule databases each having a different length and width can coexist inside the same search engine so that operating characteristics (such as speed and volume occupation) and flexibility (the coexistence of different rule databases) are improved."* The preceding text excerpt clearly indicates that groupings of differently sized rules are grouped together and stored in separate databases (e.g. memory devices).) (Column 6, Lines 21-35).

As per Claim 23, Lu discloses the interface further comprises registers configured to store the data key (i.e. Figure 2 clearly indicates that the data keys are stored in registers.) (Figure 2).

As per Claim 24, Lu discloses registers coupled to the comparator for storing the results for different rules (i.e. Figure 2 clearly indicates that the rule vectors and results are stored in registers coupled to the comparator.) (Figure 2).

As per Claim 25, Lu discloses registers coupled to the comparator for storing the results for different rules (i.e. Figure 2 clearly indicates that the rule vectors and results are stored in registers coupled to the comparator.) (Figure 2).

As per Claim 26, Lu discloses a switching unit configured to switch between modes of the system, each mode having one of a plurality of respective numbers of bits in each data key and one of a plurality of numbers of rules (i.e. *"This invention also provides a flexible and high-performance packet classification algorithm that support a plurality of rule databases or sub-tables. In addition, this invention permits the co-existence of a plurality of rule databases each having a different length and width in the same search engine. Therefore, the design can provide actual improvements (higher speed, smaller volume occupation) and flexibility (possible coexistent of different rule databases). Moreover, the invention not only can provide a dynamic setting of different rule width for sub-tables on physical memory units, but can also provide unlimited flexibility to the search algorithm. In brief, the search method of this invention can be used as a general-purpose search engine in the design of network processor or in any situation where rapid search is necessary. The search method can serve*

even as a replacement technology for CAM." The preceding text excerpt clearly indicates that the system supports dynamic switching of data key size and rule sets.) (Column 3, Lines 19-34).

As per Claim 27, Lu discloses a switching unit configured to switch between modes of the system, each mode having one of a plurality of respective numbers of bits in each data key and one of a plurality of numbers of rules (i.e. *"This invention also provides a flexible and high-performance packet classification algorithm that support a plurality of rule databases or sub-tables. In addition, this invention permits the co-existence of a plurality of rule databases each having a different length and width in the same search engine. Therefore, the design can provide actual improvements (higher speed, smaller volume occupation) and flexibility (possible coexistent of different rule databases). Moreover, the invention not only can provide a dynamic setting of different rule width for sub-tables on physical memory units, but can also provide unlimited flexibility to the search algorithm. In brief, the search method of this invention can be used as a general-purpose search engine in the design of network processor or in any situation where rapid search is necessary. The search method can serve even as a replacement technology for CAM."* The preceding text excerpt clearly indicates that the system supports dynamic switching of data key size and rule sets.) (Column 3, Lines 19-34).

As per Claim 28, Lu discloses a data switch comprising a parsing system configured to extract a data key from received packets, and a system configured to compare the extracted data key to portions of a plurality of rules defined by corresponding masks, the system including, an interface configured to divide the data key into chunks (i.e. *"Accordingly, one object of the present invention is to provide a flexible and high-performance packet classification algorithm that involves the conversion of original rule database into a rule mapping table format for storage. The rule mapping table is formed by dividing an input key into a*

plurality of sub-keys, and then sequentially comparing the ordering of each sub-key with the same sub-key field of each rule. Finally, the results of the comparison (`1` indicates a match while a `0` indicates a mismatch) are stored in the rule mapping table through bit mapping." The preceding text excerpt clearly indicates that a data key is divided into a plurality of chunks (e.g. subkeys).) (Column 2, Lines 20-31), a memory configured to receive the chunks of the data key from the interface, and for successive rules, to use the chunks of the data key and the rule as address data to extract data indicative of whether a bitwise comparison of each chunk of the data key with a corresponding portion of the mask is equal to a bitwise comparison of the corresponding portion of the mask and a corresponding portion of the rule (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120, searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1 }. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150."* The preceding text excerpt clearly indicates that a rule table, which indicates correspondence between the rules and the keys, is built by comparing the subkeys with masked portions of the rule. Examiner notes that the comparison is a bitwise AND operation.) (Column 4, Lines 33-57); and a comparator for

examining the data extracted from the memory to determine if the rules are obeyed for the entire data key (i.e. *"The goal of this invention is to provide a high-speed and economical search method particularly for searching data that includes don't care bit fields. To simplify explanation, a rule table having just five rules is selected in the following illustration. FIG. 1 is a diagram showing a packet classification database having five 8-bit rules. As shown in FIG. 1, the rule table 100 has five 8-bit rules with each rule bit having `1`, `0` or `x` (don't care). The search method of this invention is capable of finding a rule that conforms to a particular input key in rule table 100. When more than one rule conforms to the input key, the search method takes up the earlier one in the rule table list as the lookup result of the search. Using an input key #0(`111110011`) as an example, conformed rules in table 100 includes rule #1(`11110×11`), rule #2(`1×××0011` and rule #4(`××××××××`). Under such circumstances, the search method chooses rule #1 as a search result 100. The following is a description of the packet classification algorithm provided by this invention. First, the original rule table must be converted into a rule mapping table and stored in a memory unit. FIG. 2 is a diagram showing the packet classification algorithm according to this invention. The mapping table 120 shown in FIG. 2 is generated by dividing the input key into a plurality of sub-keys. Thereafter, various combinations of the bits inside each sub-key value and same sub-key value field to of each rule are compared. The result of the comparison is stored in mapping table 120 according to a bit map method (`1` represents a match while `0` represent a mismatch). Later, in the following description, rule vector (I, J) is used to refer to the converted bit mapping of mapping table 120 when the value of the Ith sub-key with sub-key value equal to J."* The preceding text excerpt clearly indicates that that data key is determined to obey the rule if all of the subkeys match each of the corresponding rule vectors. Examiner notes that this correspondence data is stored in the rule mapping table and examined to determine if keys conform to rules.) (Column 4, Lines 1-32).

As per Claim 29, Lu discloses the memory is structured two dimensionally, a first dimension corresponding to the different chunks of the data key and a second

dimension corresponding to the different rules (i.e. *"Using rule table 100 in FIG. 1 as an example, if the 8-bit input key is divided up into four 2-bit sub-keys, rule mapping table 120 shown in FIG. 2 is obtained. For sub-key #0 {bit #1, bit #0} of the input key, rule vector (0,0) that corresponds to the sub-key #0 with sub-key value equals to 0, is {1, 0, 0, 1}. This indicates that when bit #1 and bit #0 of the input key is {0,0}, rule #0, rule #3 and rule #4 conform. Similarly, rule vector (0,3) that corresponds to the sub-key #0 with sub-key value of 3 is {0, 1, 1, 1, 1}. This indicates that when bit #1 and bit #0 of the input key is {1,1}, rule #1, rule #2 #3 and rule #4 conform. After the establishment of rule mapping table 120, searching operations are very much simplified. All that is required is to extraction of all the rule vectors corresponding to the search key value. Using the input key #1(`11110011`) in FIG. 2 as an example, four rule vectors including rule vector (0, 3), rule vector (1, 0), rule vector (2, 3) and rule vector (3, 3) (shown in shade) are required. A logic AND operation of the rule bit mappings are carried out to obtain a conformed rule vector: {0, 1, 1, 0, 1 }. This indicates that the input key conforms to rule #1, rule #2 and rule #4. Finally, a AND logic element 130 and a priority encoder 140 are used to extract the leftmost rule, that is, rule #1. Hence, rule #1 becomes the lookup result 150."* The preceding text excerpt and Figure 2 clearly indicate that the mapping table and the rule vectors are stored in two separate data dimensions.) (Figure 2; Column 4, Lines 33-57).

As per Claim 30, Lu discloses the memory comprises a plurality of memory devices, each memory device corresponding to multiple chunks of the data key, such that all of the memory devices together correspond to all of the chunks of the data key (i.e. *"Furthermore, the rule mapping table can be dissected into a plurality of sub-tables such that the number of rules and rule width in each sub-table can be set. Each sub-table has an initial scan value register, a terminal scan value register and a register for recording the width of the sub-table. Each sub-table can even have a register for registering the initial address of memory for holding associated data and a register for registering size of storage location for the associated data."* The preceding text excerpt along with Figure 4 clearly indicates that the mapping tables, which hold the mappings corresponding to

multiple chunks of the data key, may be divided into subtables which contain multiple registers. Examiner notes that both the subtables and the registers may be considered to be memory devise and that all the subtables together will correspond to all of the chunks of the data key.) (Figure 4, Column3, Lines 9-17).

As per Claim 31, Lu discloses the rules are divided into groups of rules, and wherein each memory device stores the data corresponding to one of the groups of the rules (i.e. *"Through the said technique, rule mapping table lookup can be achieved with each search engine processing a portion of the sub-key fields in parallel. By using interleave matrix, the search algorithm is even capable of processing rules that have a varying width. Furthermore, this invention can support a plurality of rule databases or sub-tables. The only criteria are to set up the initial scan value, the terminal scan value and rule width of the desired search sub-table. After the setting of the initial scan value, the search engines can search for the sub-table automatically. Consequently, a plurality of rule databases each having a different length and width can coexist inside the same search engine so that operating characteristics (such as speed and volume occupation) and flexibility (the coexistence of different rule databases) are improved."* The preceding text excerpt clearly indicates that groupings of differently sized rules are grouped together and stored in separate databases (e.g. memory devices).) (Column 6, Lines 21-35).

Points of Contact

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Hicks whose telephone number is (571) 272-2670. The examiner can normally be reached on Monday - Friday 9:00a - 5:30p.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Christian Chace can be reached on (571) 272-4190. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Christian P. Chace/
Supervisory Patent Examiner, Art Unit 2165

Michael J Hicks
Art Unit 2165
Phone: (571) 272-2670
Fax: (571) 273-2670